

## *Time Protocol Design*

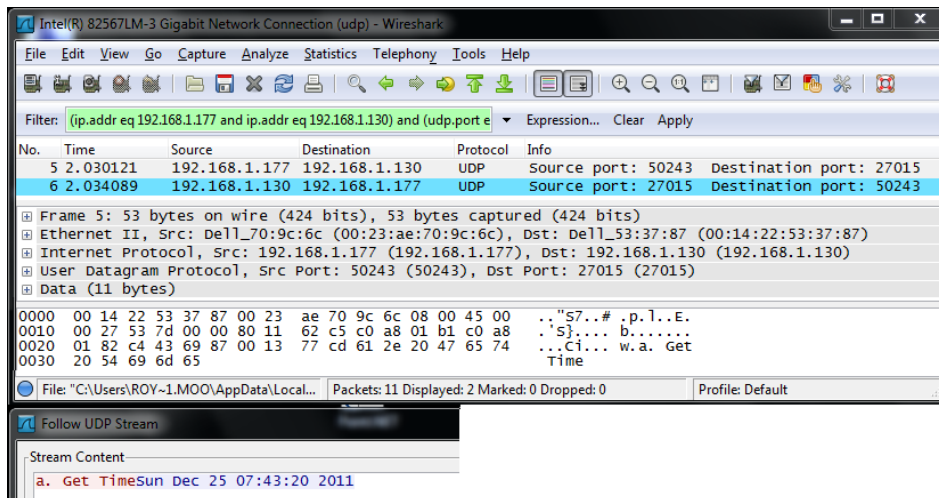
---

- In this implementation of the time protocol messages (requests and answers) are sent as Clear Text (Human Readable)
- The time client supports the following message requests:
  1. GetTime - Used to get the full current data and time.
  2. GetTimeWithoutYear – Used to get the current time only.
  3. GetTimeSinceEpoch – Used to get the current time in seconds since 1/1/1970.
  4. GetClientToServerDelayEstimation – Used to get an estimation of the Delay between client and server. (\*will be using 100 messages without waiting each time for a reply)
  5. GetDayAndMonth – Used to get the current day and month only.
  6. GetYear – Used to get the current year only.
- The time server supports the following response answers:
  1. GetTime – Reply the full current data and time in the following format:  
<Day of the week> <Month> <Day> [HH:MM:SS] <Year>
  2. GetTimeWithoutYear – Reply the current time in the following format:  
[HH:MM:SS]
  3. GetTimeSinceEpoch – Reply the current time in seconds since 1/1/1970.
  4. GetClientToServerDelayEstimation – Reply the time stamp on server at the time of the message arrival (\*using GetTickCount)
  5. GetDayAndMonth – Reply the current day and month in the following format:  
<Month> <Day>
  6. GetYear – Reply the current year only in the following format:  
[XXXX]
- Each request described in the protocol will be replied by the correct specific message.
  - All of the request will be sent by a pre-defined string
  - All of the replies will be sent according to the specified formatting

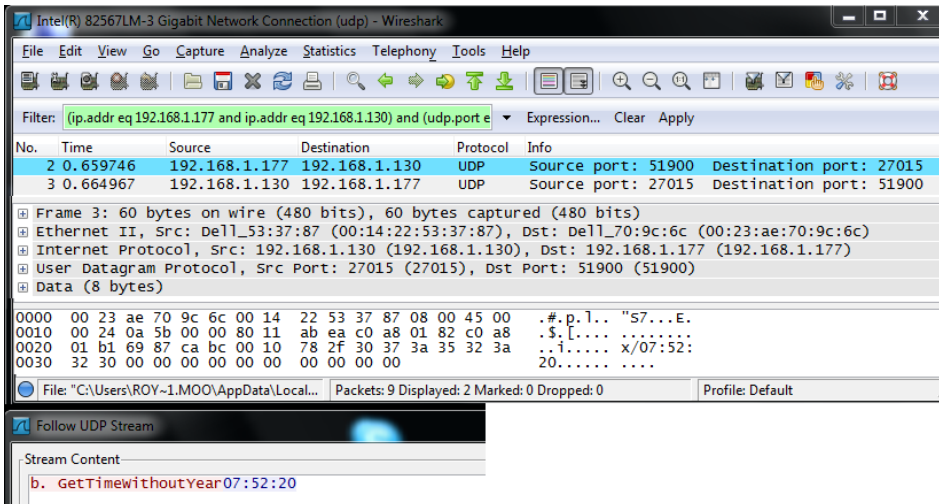
- One application for both client and server, the user selects the desired mode
- The files: “**UDPTIME.h**”, “**UDPTIME.cpp**”  
Will be Used to include all defines and dependencies needed by both the client and server instances of the application.
- The files: “**UDPTIMEClient.h**”; “**UDPCClient.cpp**” ; “**UDPTIMEServer.h**”; “**UDPServer.cpp**”  
Will be used to operate the desired mode of the application (for client/server instance)
- Client instance:
  1. The user will be prompt to enter The Server IP address (or “local” for 127.0.0.1 )
  2. The user will be prompt to enter The desired service ([switch](#))  
This will be maintained by: **SelectService**
  3. The program will then call the appropriate function to deal with this server  
(Each for every type of services...)
  4. The function will direct the program to create the appropriate message string.
  5. The function will be in charge of sending the right amount of messages and the order they will be sent (*send/receive...send/receive or send...send/receive...receive*)  
\*In some services, sending and receiving 100 messages will be needed for the client to return the needed information
- Server instance:
  1. Server will be running in an endless loop waiting for requests from client.
  2. After a string will be available by the **recvfrom** function it will be parsed by:  
**StartServer**
  3. An appropriate string will be created by the current function  
(Each for every type of services...)  
\*The designated function will be in charge of creating the right type of response string
  4. The created string will be handled by **sendBuff** and sent to the client

## Screen captures of different sessions (using Wireshark)

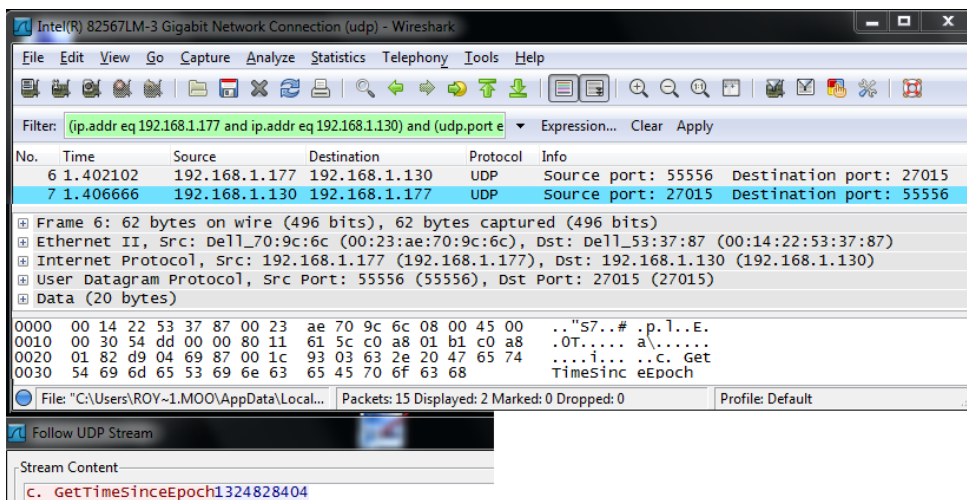
### 1. GetTime



### 2. GetTimeWithoutYear



### 3. GetTimeSinceEpoch



#### 4. GetClientToServerDelayEstimation

Intel(R) 82567LM-3 Gigabit Network Connection (udp) - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: (ip.addr eq 192.168.1.177 and ip.addr eq 192.168.1.130) and (udp.port eq 27015) Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
7	0.777158	192.168.1.177	192.168.1.130	UDP	Source port: 57493 Destination port: 27015
8	0.775462	192.168.1.177	192.168.1.130	UDP	Source port: 57493 Destination port: 27015
9	0.777299	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57493
10	0.779303	192.168.1.177	192.168.1.130	UDP	Source port: 57493 Destination port: 27015
11	0.782901	192.168.1.177	192.168.1.130	UDP	Source port: 57493 Destination port: 27015
12	0.784974	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57493
13	0.786609	192.168.1.177	192.168.1.130	UDP	Source port: 57493 Destination port: 27015
14	0.790506	192.168.1.177	192.168.1.130	UDP	Source port: 57493 Destination port: 27015
15	0.792630	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57493
16	0.795126	192.168.1.177	192.168.1.130	UDP	Source port: 57493 Destination port: 27015
17	0.798728	192.168.1.177	192.168.1.130	UDP	Source port: 57493 Destination port: 27015
18	0.800287	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57493
19	0.802432	192.168.1.177	192.168.1.130	UDP	Source port: 57493 Destination port: 27015
20	0.806326	192.168.1.177	192.168.1.130	UDP	Source port: 57493 Destination port: 27015
21	0.807965	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57493

Frame 7: 77 bytes on wire (616 bits), 77 bytes captured (616 bytes) on interface 0

Ethernet II, Src: Dell\_70:9c:6c (00:23:ae:70:9c:6c), Dst: Dell\_53:37:87 (00:14:22:53:37:87)

Internet Protocol, Src: 192.168.1.177 (192.168.1.177), Dst: 192.168.1.130 (192.168.1.130)

User Datagram Protocol, Src Port: 57493 (57493), Dst Port: 27015 (27015)

Data (35 bytes)

```

0000  00 14 22 53 37 87 00 23  ae 70 9c 6c 08 00 45 00  ..S7..#.p.l..E.
0010  00 3f 52 5e 00 00 80 11  01 0c c0 a8 01 b1 c0 a8  ?U....a.....
0020  01 82 e0 95 69 87 00 2b  6c 37 64 2e 20 47 65 74  ....i..+ 17d. Get
0030  43 6c 69 65 6e 74 54 6f  53 65 72 76 65 72 44 65  ClientTo ServerDe
0040  6c 61 79 45 73 74 69 6d  61 74 69 6f 6e          layEstim ation
  
```

File: "C:\Users\ROY~1\MOO\AppData\Local... Packets: 214 Displayed: 200 Marked: 0 Dropped: 0 Profile: Default

The image shows a network traffic analysis application window. At the top, a title bar reads "Follow UDP Stream". Below it, a section titled "Stream Content" displays a list of network events. Each event is a log entry for a "GetClientToServerDelayEstimation" operation, with a unique identifier (e.g., 939079d, 939094d, etc.). The list is scrollable. At the bottom of the window, there is a toolbar with buttons for "Find", "Save As", "Print", and a text field showing "Entire conversation (4100 bytes)". To the right of these buttons is a dropdown menu currently set to "Raw". Further right are radio buttons for different data representations: "ASCII", "EBCDIC", "Hex Dump", "C Arrays", and "Raw" (which is selected). The main area of the window is filled with the text of the stream content.

## 5. MeasureRTT

Intel(R) 82567LM-3 Gigabit Network Connection (udp) - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: (ip.addr eq 192.168.1.177 and ip.addr eq 192.168.1.130) and (udp.port eq 27015) Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.177	192.168.1.130	UDP	Source port: 57675 Destination port: 27015
3	0.005683	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57675
4	0.007822	192.168.1.177	192.168.1.130	UDP	Source port: 57675 Destination port: 27015
5	0.013106	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57675
7	0.015267	192.168.1.177	192.168.1.130	UDP	Source port: 57675 Destination port: 27015
8	0.020488	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57675
9	0.022649	192.168.1.177	192.168.1.130	UDP	Source port: 57675 Destination port: 27015
10	0.027870	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57675
11	0.030017	192.168.1.177	192.168.1.130	UDP	Source port: 57675 Destination port: 27015
12	0.035278	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57675
13	0.037477	192.168.1.177	192.168.1.130	UDP	Source port: 57675 Destination port: 27015
14	0.042661	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57675
15	0.044810	192.168.1.177	192.168.1.130	UDP	Source port: 57675 Destination port: 27015
16	0.050054	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57675
17	0.052221	192.168.1.177	192.168.1.130	UDP	Source port: 57675 Destination port: 27015
18	0.057438	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 57675

Frame 3: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)

Ethernet II, Src: Dell\_53:37:87 (00:14:22:53:37:87), Dst: Dell\_70:9c:6c (00:23:ae:70:9c:6c)

Internet Protocol, Src: 192.168.1.130 (192.168.1.130), Dst: 192.168.1.177 (192.168.1.177)

User Datagram Protocol, Src Port: 27015 (27015), Dst Port: 57675 (57675)

Data (7 bytes)

```

0000 00 23 ae 70 9c 6c 00 14 22 53 37 87 08 00 45 00  .#.p.l.. "57...E.
0010 00 23 0b 50 40 00 80 11 6a f6 c0 a8 01 82 c0 a8  .#.P@... j.....
0020 01 b1 69 87 e1 4b 00 0f 60 dd 31 31 35 39 31 31  ..i..K.. ..115911
0030 38 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .8..... ..
  
```

File: "C:\Users\ROY~1\MOO\AppData\Local... Packets: 213 Displayed: 200 Marked: 0 Dropped: 0 of : Default

Follow UDP Stream

Stream Content

```

e. MeasureRTT1159118e. MeasureRTT1159118e. MeasureRTT1159134e. MeasureRTT1159134e. MeasureRTT1159149e.
MeasureRTT1159149e. MeasureRTT1159165e. MeasureRTT1159165e. MeasureRTT1159181e. MeasureRTT1159181e.
MeasureRTT1159181e. MeasureRTT1159196e. MeasureRTT1159196e. MeasureRTT1159212e. MeasureRTT1159212e.
MeasureRTT1159227e. MeasureRTT1159227e. MeasureRTT1159243e. MeasureRTT1159243e. MeasureRTT1159259e.
MeasureRTT1159259e. MeasureRTT1159274e. MeasureRTT1159274e. MeasureRTT1159290e. MeasureRTT1159290e.
MeasureRTT1159305e. MeasureRTT1159305e. MeasureRTT1159321e. MeasureRTT1159321e. MeasureRTT1159337e.
MeasureRTT1159337e. MeasureRTT1159352e. MeasureRTT1159352e. MeasureRTT1159368e. MeasureRTT1159368e.
MeasureRTT1159383e. MeasureRTT1159383e. MeasureRTT1159399e. MeasureRTT1159399e. MeasureRTT1159415e.
MeasureRTT1159415e. MeasureRTT1159430e. MeasureRTT1159430e. MeasureRTT1159446e. MeasureRTT1159446e.
MeasureRTT1159461e. MeasureRTT1159461e. MeasureRTT1159477e. MeasureRTT1159477e. MeasureRTT1159493e.
MeasureRTT1159493e. MeasureRTT1159508e. MeasureRTT1159508e. MeasureRTT1159524e. MeasureRTT1159524e.
MeasureRTT1159539e. MeasureRTT1159539e. MeasureRTT1159555e. MeasureRTT1159555e. MeasureRTT1159571e.
MeasureRTT1159571e. MeasureRTT1159586e. MeasureRTT1159586e. MeasureRTT1159602e. MeasureRTT1159602e.
MeasureRTT1159617e. MeasureRTT1159617e. MeasureRTT1159633e. MeasureRTT1159633e. MeasureRTT1159649e.
MeasureRTT1159649e. MeasureRTT1159664e. MeasureRTT1159664e. MeasureRTT1159680e. MeasureRTT1159680e.
MeasureRTT1159695e. MeasureRTT1159695e. MeasureRTT1159711e. MeasureRTT1159711e. MeasureRTT1159727e.
MeasureRTT1159727e. MeasureRTT1159742e. MeasureRTT1159742e. MeasureRTT1159758e. MeasureRTT1159758e.
MeasureRTT1159773e. MeasureRTT1159773e. MeasureRTT1159789e. MeasureRTT1159789e. MeasureRTT1159805e.
  
```

Find Save As Print Entire conversation (2000 bytes) ASCII EBCDIC Hex Dump C Arrays Raw

Help Filter Out This Stream Close

\*To calculate to average RTT we used the same request as in 4 (GetClientToServerDelayEstimation)

The request is sent 100 times: send/wait for reply/send/wait for reply...

**The calculation of the average RTT isn't a part of the original protocol**

(It's only a specific utilization of a specific type of request)

## 6. GetDayAndMonth

Intel(R) 82567LM-3 Gigabit Network Connection (udp) - Wireshark

Filter: (ip.addr eq 192.168.1.177 and ip.addr eq 192.168.1.130) and (udp.port eq 27015)

No.	Time	Source	Destination	Protocol	Info
11	0.940877	192.168.1.177	192.168.1.130	UDP	Source port: 55035 Destination port: 27015
12	0.945242	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 55035

Frame 11: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)

Ethernet II, Src: Dell\_70:9c:6c (00:23:ae:70:9c:6c), Dst: Dell\_53:37:87 (00:14:22:53:37:87)

Internet Protocol, Src: 192.168.1.177 (192.168.1.177), Dst: 192.168.1.130 (192.168.1.130)

User Datagram Protocol, Src Port: 55035 (55035), Dst Port: 27015 (27015)

Data (18 bytes)

0000 00 14 22 53 37 87 00 23 ae 70 9c 6c 08 00 45 00 ..S7..#.p.l..E.  
0010 00 2e 57 55 00 00 00 11 5e e6 c0 a8 01 b1 c0 a8 ..WU...^.....  
0020 01 82 d6 fb 69 87 00 1a eb 7b 66 2e 20 47 65 74 ....i...{f. Get  
0030 44 61 79 41 6e 64 d4 6f 75 6e 74 68 DayAndMonth

File: "C:\Users\ROY~1.MOO\AppData\Local..." Packets: 19 Displayed: 2 Marked: 0 Dropped: 0 Profile: Default

Follow UDP Stream

Stream Content

f. GetDayAndMonthSun Dec

## 7. GetYear

Intel(R) 82567LM-3 Gigabit Network Connection (udp) - Wireshark

Filter: (ip.addr eq 192.168.1.177 and ip.addr eq 192.168.1.130) and (udp.port eq 27015)

No.	Time	Source	Destination	Protocol	Info
2	1.315045	192.168.1.177	192.168.1.130	UDP	Source port: 64859 Destination port: 27015
3	1.318964	192.168.1.130	192.168.1.177	UDP	Source port: 27015 Destination port: 64859

Frame 3: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)

Ethernet II, Src: Dell\_53:37:87 (00:14:22:53:37:87), Dst: Dell\_70:9c:6c (00:23:ae:70:9c:6c)

Internet Protocol, Src: 192.168.1.130 (192.168.1.130), Dst: 192.168.1.177 (192.168.1.177)

User Datagram Protocol, Src Port: 27015 (27015), Dst Port: 64859 (64859)

Data (4 bytes)

0000 00 23 ae 70 9c 6c 00 14 22 53 37 87 08 00 45 00 ..#.p.l.. "S7...E.  
0010 00 20 0b cc 40 00 80 11 6a 7d c0 a8 01 82 c0 a8 ...@... j}.....  
0020 01 b1 69 87 fd 5b 00 0c b1 0d 32 30 31 31 00 00 ..i...[. ..2011..  
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .....

File: "C:\Users\ROY~1.MOO\AppData\Local..." Packets: 7 Displayed: 2 Marked: 0 Dropped: 0 Profile: Default

Follow UDP Stream

Stream Content

g. Get Year2011



- הריצו את התרגיל ובקשו בקשות `GetClientToServerDelayEstimation` רבות .  
האם ייתכן שהלקוח או השרת "ייתקעו" אם לא, נמקו .  
אם כן, נמקו והציעו תיקון בתוכנית שלכם בכדי שהלקוח והשרת לא "ייתקעו".
  - מבחינת ההשחיות השונות שאנו כבר מכירים (אותן 4 השחיות שנלמדו בכיתה), מה מבטא הגודל הממוצע אותו הלקוח מחשב על סמך תשובות השרת בבקשת `GetClientToServerDelayEstimation` ?  
(המלצה: נתחו גודל זה מבחינה תאורטית ולא דווקא על סמך התוצאות המתקבלות מההרצה בפועל שעלויות להטעות, בפרט כאשר מבוצעות על אותו המחשב)
  - יתכן בהחלט כי גם השרת וגם הלקוח יתקעו.  
הדבר נובע מתצורת העבודה אותה בחרנו ליישם בתרגיל : `blocking`  
בהחלט יתכן כי באופן תיאורטי כל הבקשות של הלקוח יאבדו בדרך ואז השרת ימתין/יתקע עד לקבלת המידע האבוד ... (התיאור הזה גם לגבי הלקוח)
- בדיקה בפועל מוכיחה כי אכן יש תקיעות:

- הפתרון המתבקש לבעיה הוא לשנות את תצורת העבודה של ה `socket` לתצורת `non-blocking` (כלומר יש דגימה של החוץ בכל פרק זמן כדי לקרוא ונמנע המצב של התקיעה)
- ההשחיה שאנו מודדים בסעיף `Estimated Delay` מבטאת את `Propagation Delay` על הקשר שבין השרת ללקוח, דרך נוספת להסתכל על זה (כמו שלמדנו) היא כעל הזמן שלוקח למידע לעבור ב"צינור" מצד בלקוח לצד השרת.